FIG. 1
PRIOR ART

```
1   c:\collections
2       notes.txt
3       myletter.doc
4       c-myhomepage
5
6         s
7           homepage.html
8           myphoto.jpg
```

FIG. 2

```
1   c:\collections
2       notes.txt
3       myletter.doc
```

100

```
4       c-myhomepage
5         cspec
6         s
7           homepage.html
8           myphoto.jpg
```

FIG. 3

102

```
1   collection        c-myhomepage
2   coll-type         cf-web-page
3   coll-desc         A sample homepage collection
4   end-collection
```

FIG. 4

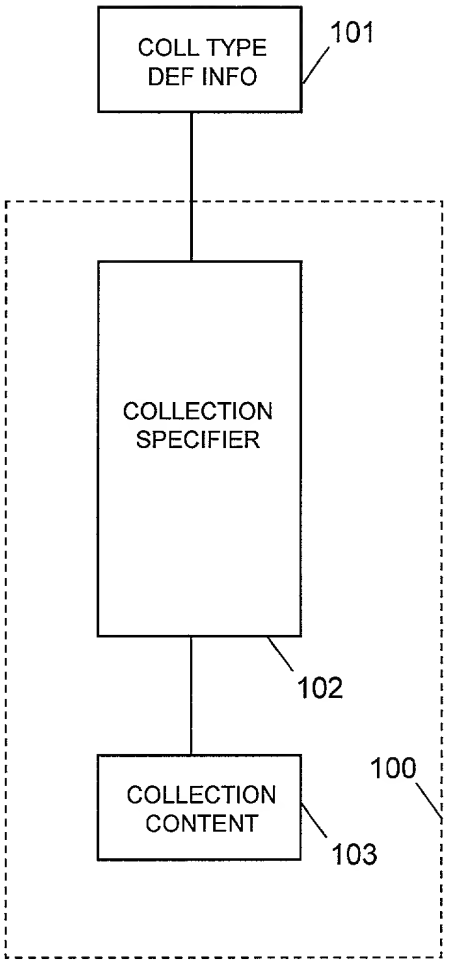

COLL TYPE DEF INFO
101
COLLECTION SPECIFIER
102
COLLECTION CONTENT
103
100

FIG. 5

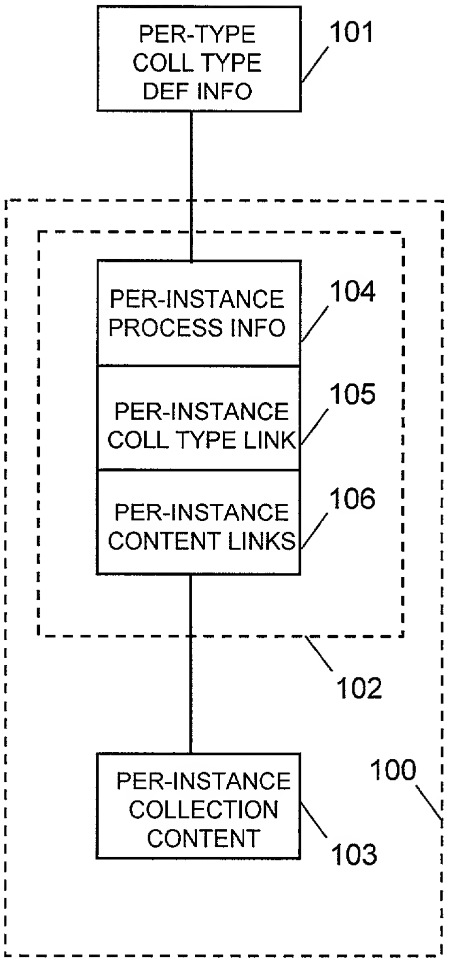
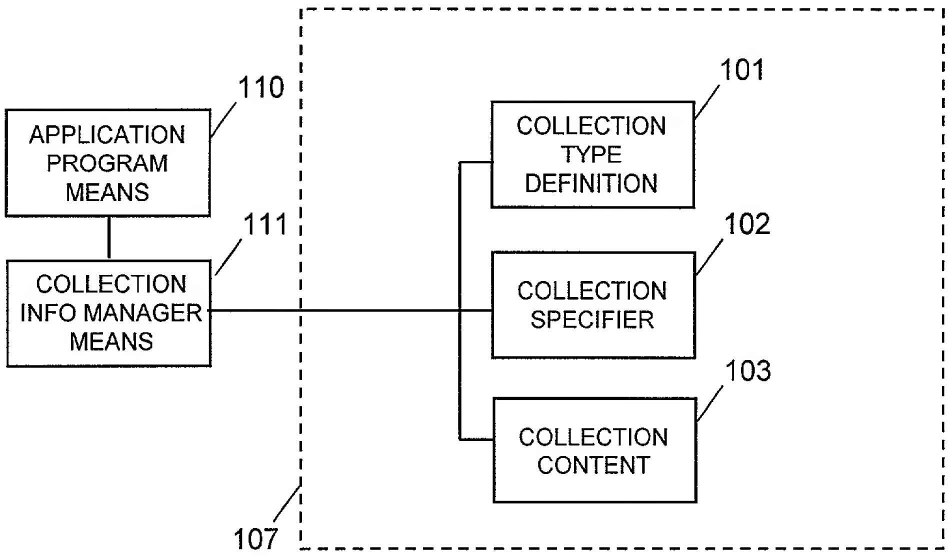
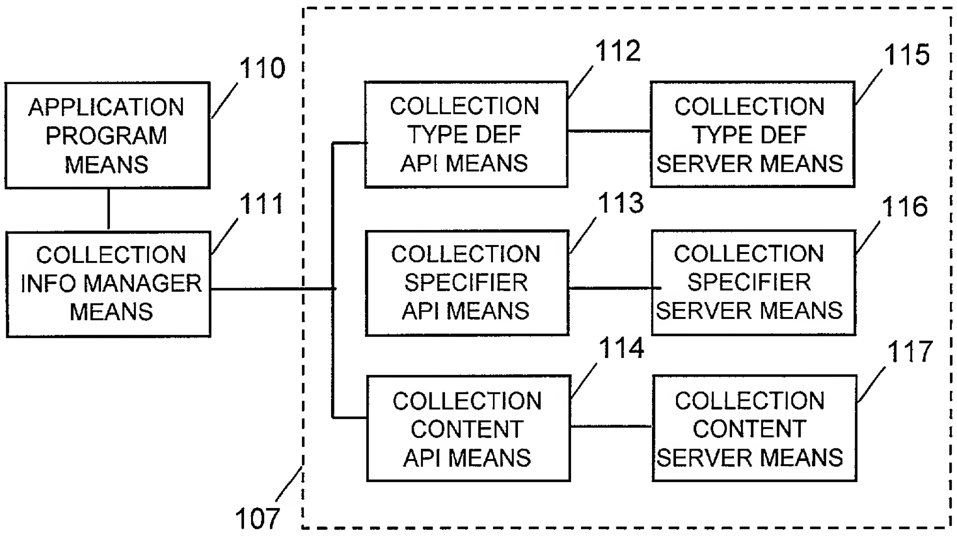

PER-TYPE COLL TYPE DEF INFO
101
PER-INSTANCE PROCESS INFO
104
PER-INSTANCE COLL TYPE LINK
105
PER-INSTANCE CONTENT LINKS
106
102
PER-INSTANCE COLLECTION CONTENT
103
100

FIG. 6



110
APPLICATION PROGRAM MEANS
111
COLLECTION INFO MANAGER MEANS
101
COLLECTION TYPE DEFINITION
102
COLLECTION SPECIFIER
103
COLLECTION CONTENT
107

FIG. 7



110
APPLICATION PROGRAM MEANS
111
COLLECTION INFO MANAGER MEANS
112
COLLECTION TYPE DEF API MEANS
115
COLLECTION TYPE DEF SERVER MEANS
113
COLLECTION SPECIFIER API MEANS
116
COLLECTION SPECIFIER SERVER MEANS
114
COLLECTION CONTENT API MEANS
117
COLLECTION CONTENT SERVER MEANS
107

FIG. 8

```
/* collection data structure */
collection-info {

  + specifier_info
      + coll-type-indicator
      + other specifier information ...

  + content_info
      + content_location_info ...
      + content_members ...
      + other content information...

  + other collection structure information...
}
```

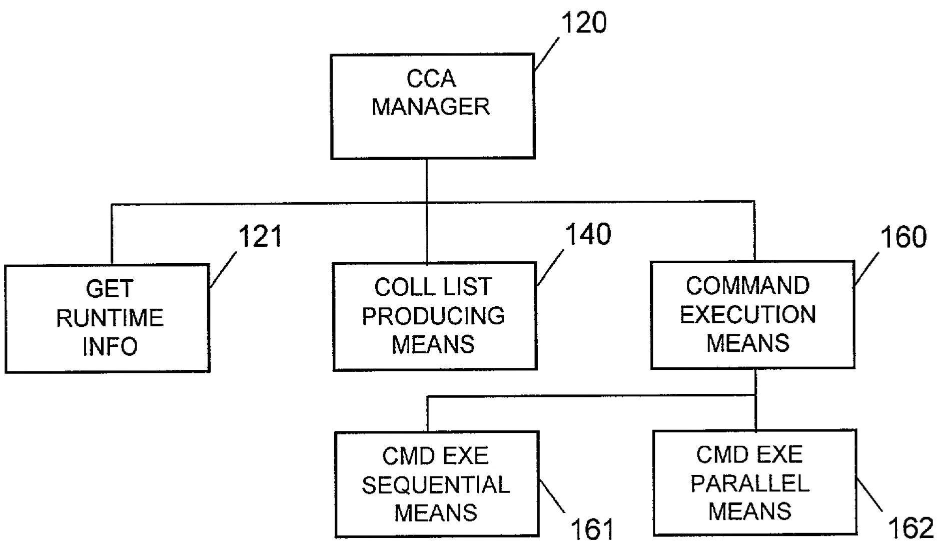
FIG. 9

```
/* collection type definition data structure */
collection-type-definition-info {

  + coll-type-name
  + collection internal structure info ...
  + collection content location info ...
  + collection content type recognition info ...

  + other collection type definition information...
}
```

FIG. 10

```
KEY                   VALUE

/* collection type internal structure definitions */
dir_source_files      .\s
dir_doc_files         .\doc

/* content location definitions (per-type content links) */
content_subtree_http    http://host.com/some/dir/name
content_subtree_ftp     ftp://host.com/some/dir/name
content_subtree_nfs     /some/local/directory/name

/* content type recognition definitions */
content_policy        subtree_below_cspec_file
content_file_type     .c      file_cpp
content_file_type     .c      file_c
content_file_type     .h      file_c_include
content_file_type     .doc    file_ms_word
content_file_type     .html   file_html
content_file_type     .xls    file_ms_excel

/* collection processing definitions */
compile_c_files       yes
compiler_windows      vc++
compiler_unix         gcc
build platforms       Win98, Win2000, linux
process files         compile link
link libraries        stdio math sock

/* results dispatching definitions */
results_ftp_host      ftp.output.com
results_ftp_dir       c:\ftphome\collection\results
```

FIG. 11



120
CCA MANAGER
121
GET RUNTIME INFO
140
COLL LIST PRODUCING MEANS
160
COMMAND EXECUTION MEANS
CMD EXE SEQUENTIAL MEANS
161
CMD EXE PARALLEL MEANS
162

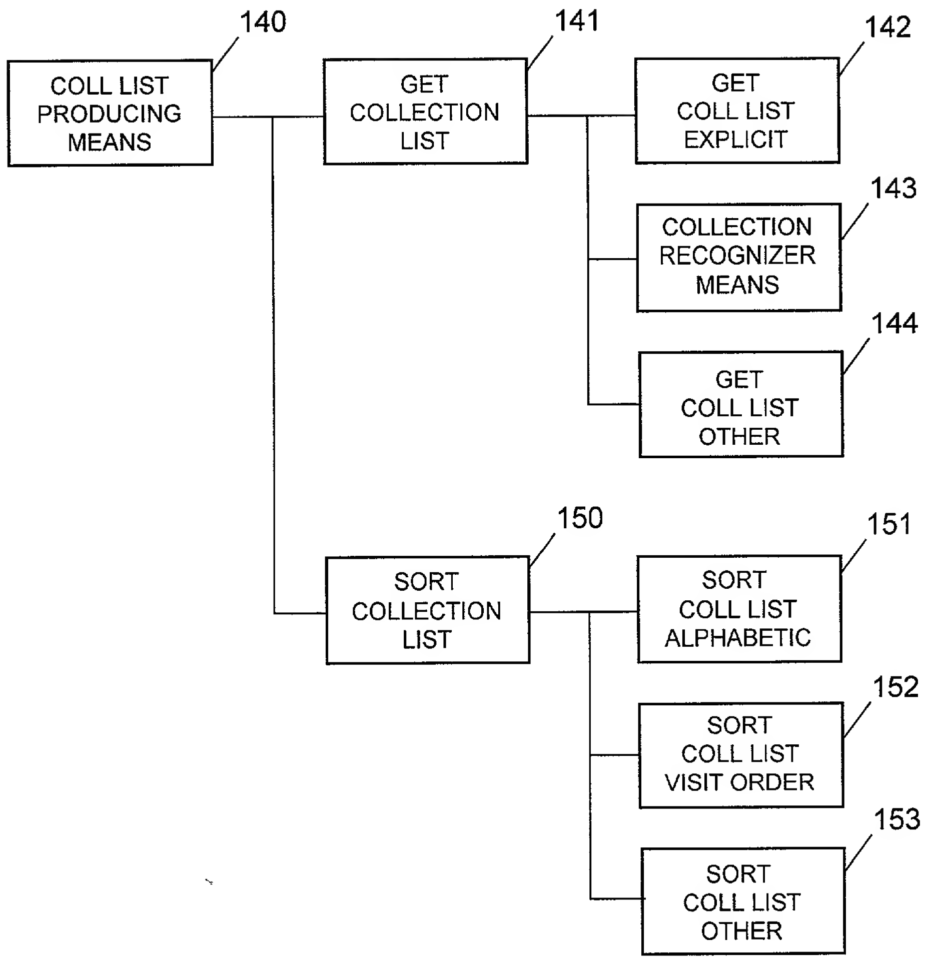
FIG. 12

```
/* simplified cca algorithm */

Call get runtime information to obtain a list of
commands to apply

Call collection list producing means to obtain a
list of target collections for command application

Call command execution means to apply commands
to target collections
```

FIG. 13

```
/* runtime information data structure */
runtime-info {

  + invocation options
  + runtime environment information
  + commands to execute on recognized collections
  + collection recognition criteria

  ...
}
```

FIG. 14



140
COLL LIST PRODUCING MEANS
141
GET COLLECTION LIST
142
GET COLL LIST EXPLICIT
143
COLLECTION RECOGNIZER MEANS
144
GET COLL LIST OTHER
150
SORT COLLECTION LIST
151
SORT COLL LIST ALPHABETIC
152
SORT COLL LIST VISIT ORDER
153
SORT COLL LIST OTHER

FIG. 15

```
/* simplified algorithm for list producing means */
Build data structures

/* Obtain list of target collections for command application */
- add explict invocation-provided collections to list
- add collection-recognizer-provided collections to list
- add collections from other sources to list

/* Build sorted-colls data structures */
- create list of target collections sorted alphabetically
- create list of target collections sorted by visit order
- create list of target collections sorted by other means

Return completed coll-list-info data structure to caller
```

FIG. 16

```
/* list of target collections for command application */
target-coll-list {

  + list of collection-structures (eg FIG 8)

}
```

FIG. 17

```
/* structure for holding sorted lists of collections */
sorted-colls {

  + sort-type = ALPHA, VISIT_ORDER, ...
  + sorted-collection-set {
      + coll-1 in sort order
      + coll-2 in sort order
      + ...

}
```

FIG. 18

```
/* structure for holding collection list info */
coll-list-prod-info {

  + list of collection structures (target-coll-list)
  + list of collection type definition structures (FIG 9)
  + list of sorted-colls structures
  + collection recognition info ...

}
```

FIG. 19

```
1     c:\collections
2         programs
3           helloworld
4             c-hello-library
5             c-hello

6         c-myprogram

7         parts
8           c-include-files
9           c-library-one
10          c-library-two

11        webstuff
12          c-myhomepage

13        c-myphotos
```

FIG. 20

```
0     /* File colls-fig-20.txt holds list of collections for tree FIG 19 */
1     c:\collections\programs\helloworld\c-hello-library
2     c:\collections\programs\helloworld\c-hello
3     c:\collections\c-myprogram
4     c:\collections\parts\c-include-files
5     c:\collections\parts\c-library-one
6     c:\collections\parts\c-library-two
7     c:\collections\webstuff\c-myhomepage
8     c:\collections\c-myphotos
```

FIG. 21

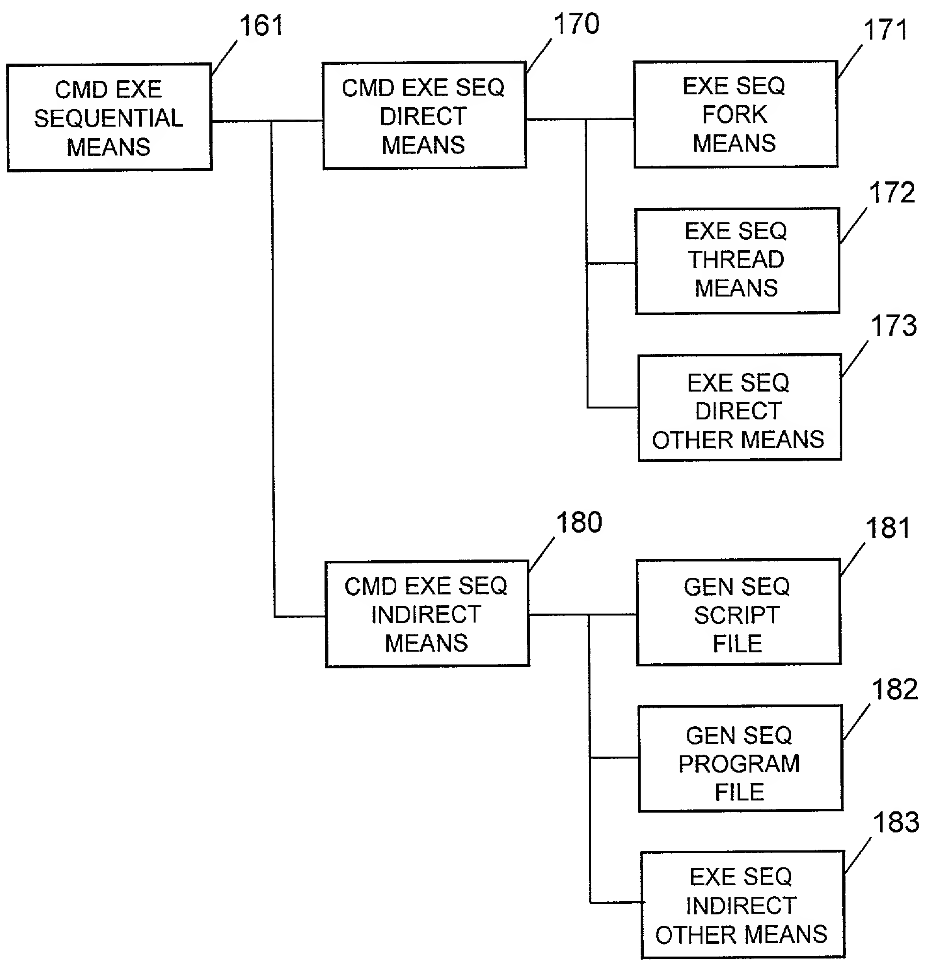

161
CMD EXE SEQUENTIAL MEANS
170
CMD EXE SEQ DIRECT MEANS
171
EXE SEQ FORK MEANS
172
EXE SEQ THREAD MEANS
173
EXE SEQ DIRECT OTHER MEANS
180
CMD EXE SEQ INDIRECT MEANS
181
GEN SEQ SCRIPT FILE
182
GEN SEQ PROGRAM FILE
183
EXE SEQ INDIRECT OTHER MEANS

FIG. 22

```
/* simplified algorithm for execute sequential direct  means */
Build data structures

/* walk list of target collections */
For each coll in list of one-coll-cmd-exe structures
  - change working directory to desired execution directory

/* execute direct commands on each target collection */
  - for each command in list of cmd-exe-status structures
        - execute the command using a subordinate helper module
        - record command status and errors
        - if command status = FAILS continue with next collection
        - continue with next command in list of commands

/* clean up and return status */
Change working directory back to original invocation directory
Return overall execution status to caller
```

FIG. 23

```
/* command and status info for 1 command */
cmd-exe-status {

  + command to execute
  + return code status of execution attempt
  + other error information
  + ...

}
```

FIG. 24

```
/* N commands and status info for 1 target collection */
one-coll-cmd-exe {

  + target collection for commands
  + list of cmd-exe-status structures
  + ...

}
```

FIG. 25

```
/* commands and status info for all target collections */
all-coll-cmd-exe {

  + list of coll-cmd-exe structures
  + execution-type = DIRECT, INDIRECT
  + exe-direct-method = FORK, THREAD, ...
  + exe-indirect-method = BATCH, PERL, ...

}
```

FIG. 26

```
if not ()==(%1) goto goodargs
echo Usage:  doinseq command1 c2 c3 ... c5
echo         doinseq  make all
echo         doinseq  dir *.c
goto quit

:goodargs

cd c:\collections\c-myphotos\win98.plt
call %1 %2 %3 %4 %5 %6 %7 %8 %9
cd c:\collections

cd c:\collections\parts\c-library-one\win98.plt
call %1 %2 %3 %4 %5 %6 %7 %8 %9
cd c:\collections

cd c:\collections\programs\helloworld\c-hello\win98.plt
call %1 %2 %3 %4 %5 %6 %7 %8 %9
cd c:\collections
...
:quit
```

FIG. 27

```
C:\> cca doinseq –explicit-colls colls-fig-20.txt –platform win98.plt

C:\> doinseq <a-command-to-execute>
C:\> doinseq ls
C:\> doinseq make all
```

FIG. 28

```
/* simplified algorithm for execute sequential indirect */

Build data structures
Generate a batch file template framework to hold commands

/* walk list of collections and generate commands into the batch file */
For each coll in list of coll-cmd-exe structures

    - emit command to change working directory into
      desired execution directory
    - emit script file argument variables to hold commands that
      are passed in to the script file for execution
    - emit command to change back to original working directory
    - continue with next collection in list of collections

Return batch file execution status
```

FIG. 29

```
/* rec-coll recognized-collections data structure */

rec-coll {
  + rec-coll-list

      + coll-structure-1
          + cspec_info ...
          + ctype_def_info ...
          + ccontent_info ...
          + other_coll_info

      + coll-structure-2
          + cspec_info ...
          + ctype_def_info ...
          + ccontent_info ...
          + other_coll_info

      + coll-structure-3
          ...

  + other collection recognition info
}
```

FIG. 30

```
1     c:\collections
2         programs
3           helloworld

4               c-hello           ctype = cf-program
                                  default vo=100
5               c-hello-library   ctype = cf-library
                                  default vo=50


6         c-myprogram             ctype = cf-program
                                  default vo=100


7         parts
8           c-include-files       ctype = cf-includes
                                  default vo=10
9           c-library-one         ctype = cf-library
                                  default vo=10
10          c-library-two         ctype = cf-library
                                  EXPLICIT vo=49


11        webstuff
12          c-myhomepage          ctype = cf-doc-html
                                  default vo=100


13        c-myphotos              ctype = cf-web-page
                                  default vo=100
```

FIG. 31

	Collection Type Name	Visit Order Ranking
1	cf-initial	10
2	cf-library	50
3	cf-program	100
4	cf-web-page	100
5	cf-doc-sgml	100
6	cf-doc-html	100

FIG. 32

1	collection	c-library-two
2	coll_type	cf-library
3	coll_desc	A library with explicit visit order
4	coll-visit-order	49
5	end-collection	

FIG. 33

1	c-hello	100
2	c-hello-library	50
3	c-myprogram	100
4	c-library-one	50
5	c-library-two	49
6	c-include-files	10
7	c-myhomepage	100
8	c-myphotos	100

FIG. 34

```
/* simplified visit order algorithm */
Receive unsorted list of collections

Obtain numeric visit order values for each collection in list
Sort the list of collections according to execution visit order

Write sorted information to sorted-colls data structure (FIG 17)
Return sorted-colls data structure to calling module
```

FIG. 35

1	c-include-files	10
2	c-library-two	49
3	c-library-one	50
4	c-hello-library	50
5	c-hello	100
6	c-myphotos	100
7	c-myhomepage	100
8	c-myprogram	100

FIG. 36

```
if not ()==(%1) goto goodargs
echo Usage:   doinseq   command1 c2 c3 ... c5
echo          doinseq   make all
echo          doinseq   dir *.c
goto quit

:goodargs

cd c:\collections\parts\c-include-files\win98.plt   (vo=10)
call %1 %2 %3 %4 %5 %6 %7 %8 %9
cd c:\collections

cd c:\collections\parts\c-library-two\win98.plt     (vo=49)
call %1 %2 %3 %4 %5 %6 %7 %8 %9
cd c:\collections

cd c:\collections\parts\c-library-one\win98.plt     (vo=50)
call %1 %2 %3 %4 %5 %6 %7 %8 %9
cd c:\collections

cd c:\collections\programs\helloworld\c-hello-library\win98.plt
call %1 %2 %3 %4 %5 %6 %7 %8 %9             (vo-50)
cd c:\collections

cd c:\collections\programs\helloworld\c-hello\win98.plt
call %1 %2 %3 %4 %5 %6 %7 %8 %9             (vo=100)
cd c:\collections
...

:quit
```

FIG. 37

	Visit Order Set Name	Visit Order Definition File
1	vo-software	vo-software.def
2	vo-doc	vo-doc.def
3	vo-xxx-name	vo-xxx-name.def

FIG. 38

0	vo-software.def:	
1	cf-initial	10
2	cf-library	50
3	cf-program	100
4	cf-web-page	100
5	cf-doc-sgml	100
6	cf-doc-html	100

FIG. 39

0	vo-doc.def:	
1	cf-doc-sgml	10
2	cf-doc-html	10
3	cf-doc-indexes	20

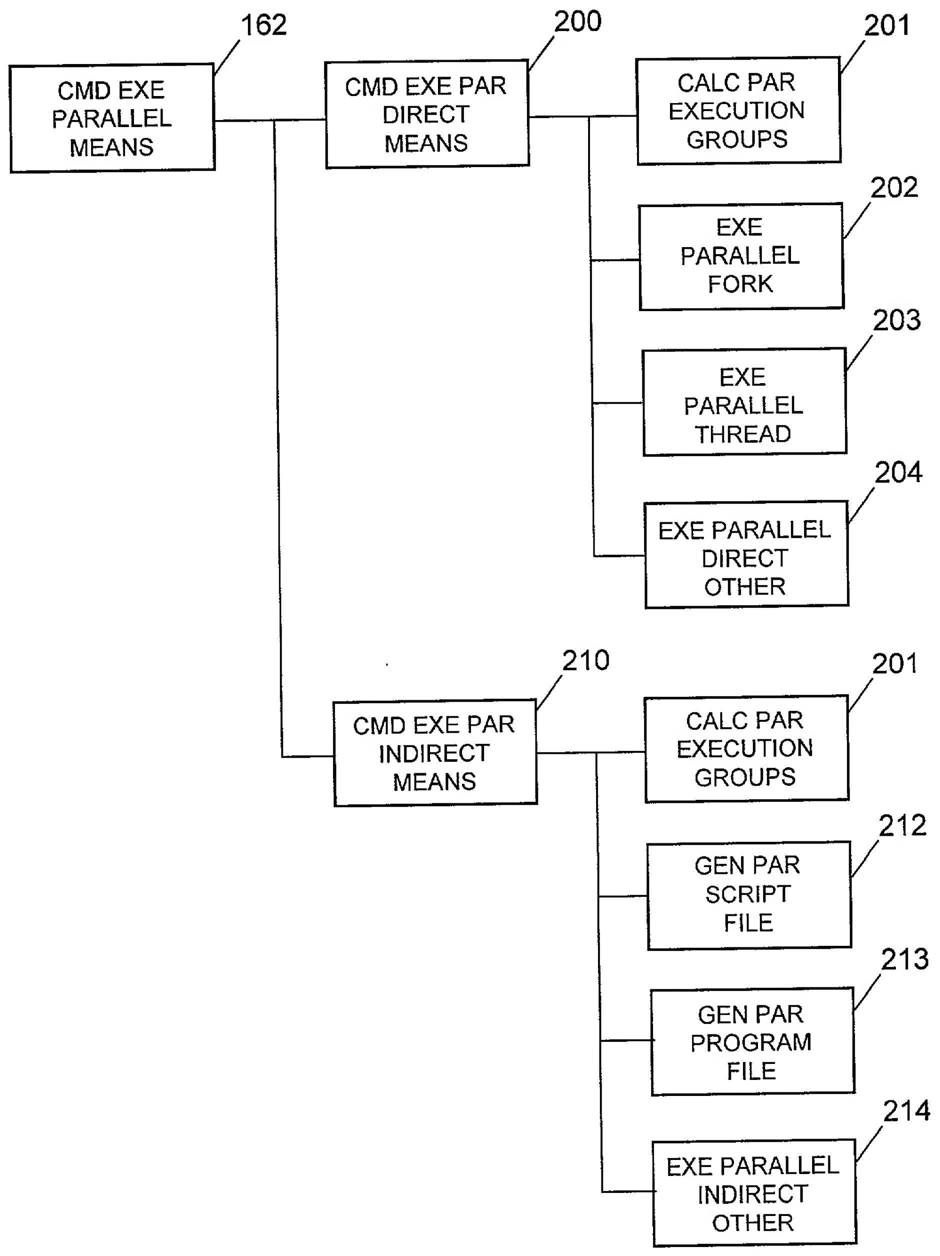
FIG. 40

```
collection          c-program-doc
coll_type           cf-doc-sgml
coll_desc           A doc with multiple explict visit orders
coll-visit-order    vo-software 49
coll-visit-order    vo-doc       10
end-collection
```

FIG. 41

```
/* simplified algorithm for calculating parallel execution groups */
Obtain list of target collections, sorted into proper visit order
Obtain physical parallelism limit = phys_par_limit
Obtain administrative parallelism limit = admin_limit

/* calculate problem parallelism limit */
set problem_par_limit = 1
For each unique visit order in list of target collections,
  - count number of collections with current visit order value
  - if current_count > problem_par_limit,
      set problem_par_limit = current_count

/* calc min of problem, physical, and admin parallelism limits */
useful_par_limit = min (problem_par_limit, physical_par_limit,
                        admin_par_limit)

/* calc parallel execution groups using useful_par_limit */
For each unique visit order in list of target collections,
  - create a new parallel execution group
    containing all collections that match the current visit order
  - if group size is > useful_par_limit,
    split group into smaller groups
    until no groups exceed useful_par_limit
  - continue with next unique visit order in list of target collections

Return parallel execution ordering and limits to caller
```

FIG.42



162
CMD EXE PARALLEL MEANS
200
CMD EXE PAR DIRECT MEANS
201
CALC PAR EXECUTION GROUPS
202
EXE PARALLEL FORK
203
EXE PARALLEL THREAD
204
EXE PARALLEL DIRECT OTHER
210
CMD EXE PAR INDIRECT MEANS
201
CALC PAR EXECUTION GROUPS
212
GEN PAR SCRIPT FILE
213
GEN PAR PROGRAM FILE
214
EXE PARALLEL INDIRECT OTHER

FIG. 43

```
1  /* simplified algorithm for execute parallel direct */
2  Build data structures

3  /* calculate parallel execution groups */
4  Calculate parallel execution groups per algorithm FIG 42

5  /* execute collections in each parallel group in parallel */
6  For each parallel execution group, in proper parallel exe order {
7    - Execute in parallel {
8      - For each collection in the parallel exe group
9        - change working directory to desired execution directory

10 /* execute commands for one collection */
11       - for each command in list of cmd-exe-status structures
12         - execute the command
13         - record command status and errors
14         - if command status = FAILS continue with next collection
15         - continue with next command in list of commands

16   - Wait for all collections in the group to finish
17   } /* end of parallel section beginning on Line 7
18 } /* end of parallel execution group list traversal beginning on Line 6

19 Return overall execution status information to caller
```

FIG. 44

```
/* collections for 1 parallel execution group */
parallel-exe-group {
  + par-exe-rank = 1, 2, 3, ...
  + par-exe-group-coll-list {
      + collection-1
      + collection-2
      + ...
}
```

FIG. 45

```
/* collections for all parallel execution groups */
cmd-exe-parallel {

   + list of coll-cmd-exe structures
   + list of parallel-exe-group structures
   + exe-type = DIRECT, INDIRECT
   + exe-method-direct = FORK, THREAD, ...
   + exe-method-indirect = PERL, PROGRAM,

}
```

FIG. 46

```
time=0,  vo=10        c-include-files

time=1,  vo=49        c-library-two

time=2,  vo=50        c-library-one
                      c-hello-library

time=3,  vo=100       c-hello
                      c-myprogram
                      c-myhomepage
                      c-myphotos
```

FIG. 47

```
1  /* simplified algorithm for execute parallel indirect */

2  Build data structures

3  /* calculate parallel execution groups */
4  Calculate parallel execution groups per algorithm FIG 42

5  /* execute collections in each parallel group in parallel */
6  For each parallel execution group, in proper parallel exe order {
7    - Emit commands for executing in parallel {
8      - For each collection in the parallel exe group
9        - emit cd command  to change to desired execution directory

10 /* emit execution commands for one collection */
11       - for each command in list of cmd-exe-status structures
12         - emit syntax for the command to be executed in parallel
13         - emit syntax to record command status and errors
14         - continue with next command in list of commands

15   - Emit syntax to wait for all collections in the group to finish
16   }  /* end of parallel section beginning on Line 7
17 } /* end of parallel execution group list traversal beginning on Line 6

18 Return overall execution status information to caller
```

FIG. 48

```
#!/bin/sh
if [ $# -lt 1 ] ; then
      echo "Usage:  doinparallel command-1 c-2 c-3 ... c-N"
      echo "            doinparallel copy file1 file2"
      echo "            doinparallel make all"
      exit 1
fi

# at time 0, apply parallel execution group #1
cd /collections/parts/c-include-files/linux.plt
$@
cd /collections

...  # execution group #2 omitted to save space

# at time 2, apply commands parallel execution group #3
cd /collections/parts/c-library-one/linux.plt
$@ &
cd /collections/programs/helloworld/c-hello-library/linux.plt
$@ &
# wait for all parallel jobs to complete
wait
cd /collections

# at time 3, apply commands to parallel execution group #4
cd /collections/programs/helloworld/c-hello/linux.plt
$@ &
cd /collections/c-myprogram/linux.plt
$@ &
cd /collections/webstuff/c-myhomepage/linux.plt
$@ &
cd /collections/c-myphotos/linux.plt
$@ &
# wait for all parallel jobs to complete
wait
cd /collections

exit 0
```

FIG. 49

```
 1  c:\collections
 2    programs
 3      helloworld
 4        c-hello
 5          cspec
 6          s...
 7          win98.plt...
 8          gnulinux.plt...

 9    c-myprogram
10      cspec
11      s...
12      win98.plt...
13      gnulinux.plt...

14    parts
15      c-include-files
16        cspec
17        s...
18        win98.plt...
19        gnulinux.plt...
```

FIG. 50

```
1 Visit all s dirs beside cspec files, non-recursively
2 Visit all immediate child dirs, recursively
3 Format files in all s dirs   (visit s dirs)
4 Modify all cspec files       (visit root dirs of collections)
5 Delete all collections       (visit parent dirs of collections)
6 Clean up win98.plt dirs      (visit win98.plt dirs)
7 Delete all plt dirs          (visit root dirs of collections)
```